Silent Threshold Cryptography from Pairings

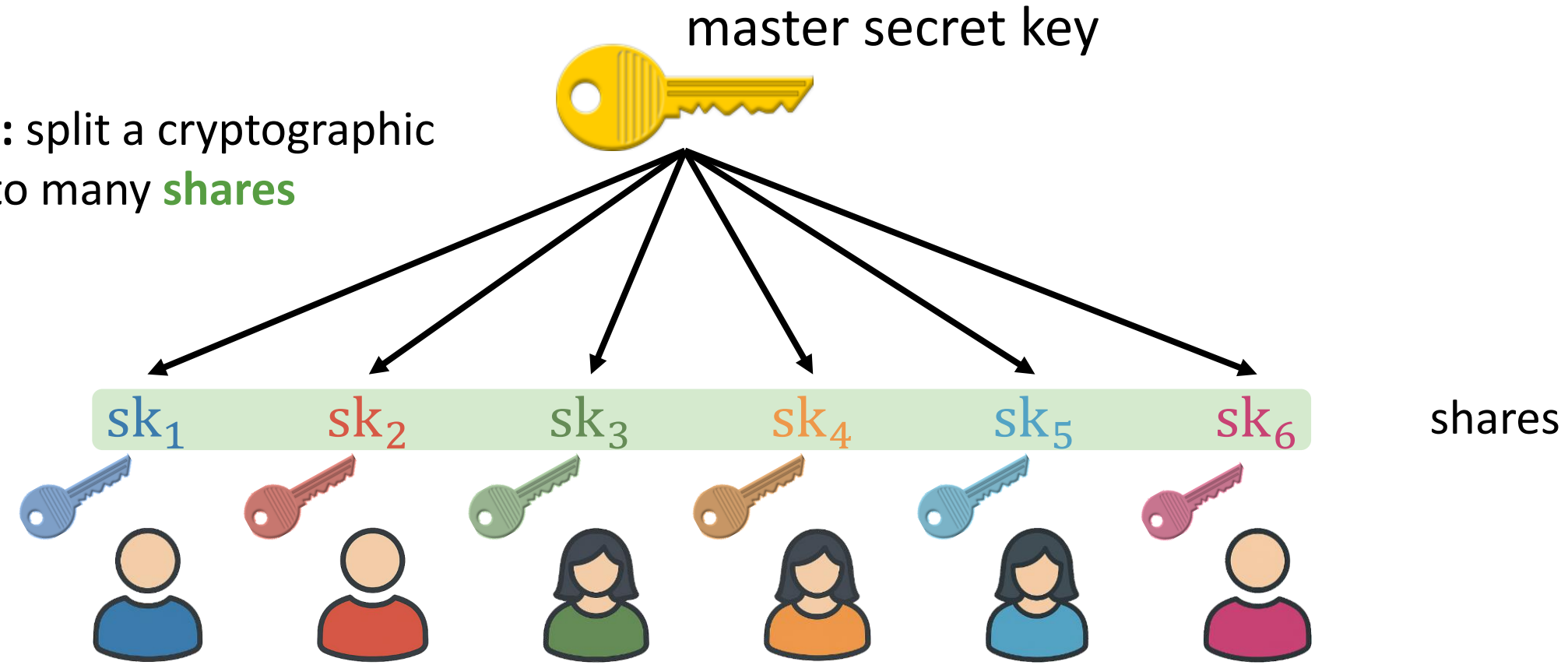
Brent Waters and **David Wu**

May 2026

Threshold Cryptography

[Des87, Fra89, DF89]

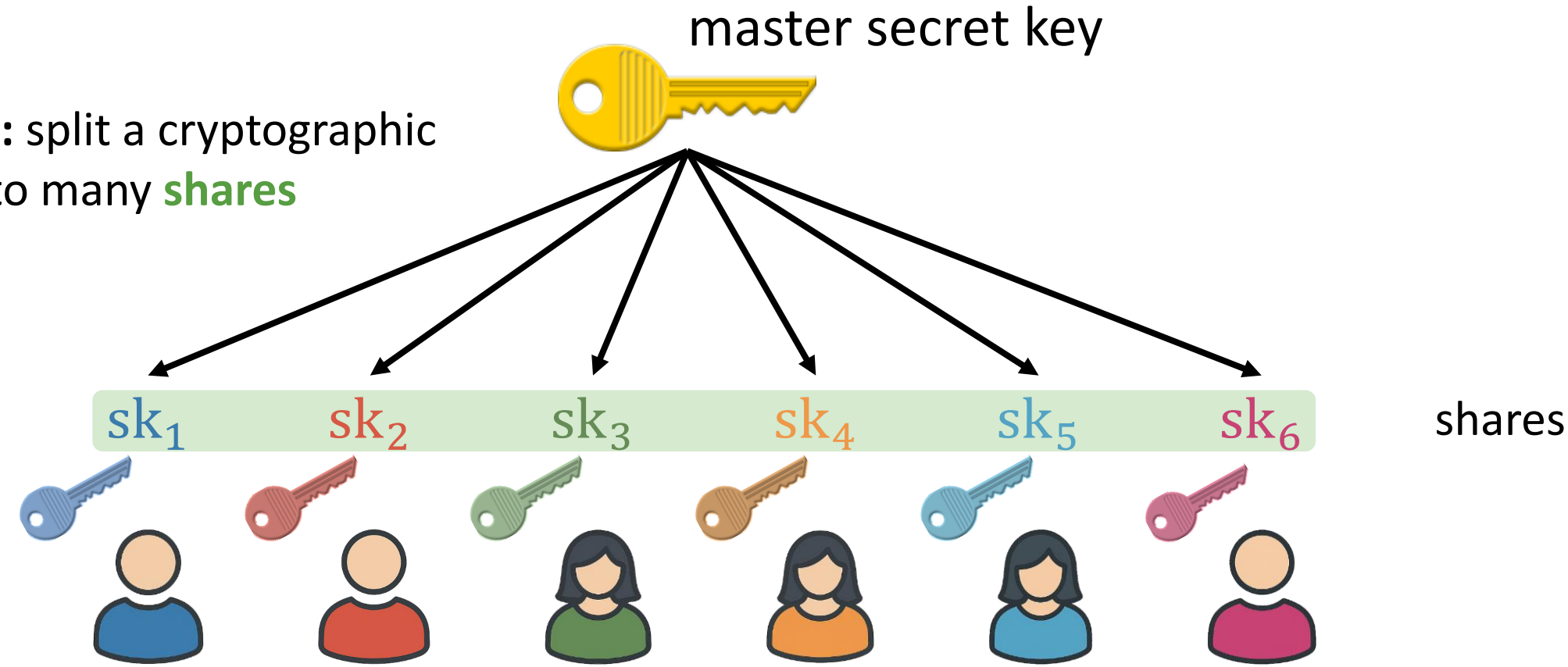
Typical setup: split a cryptographic key into many **shares**



Only an **authorized set** of parties can perform a target action (e.g., signing, decryption, etc.)

Who Generates the Shares?

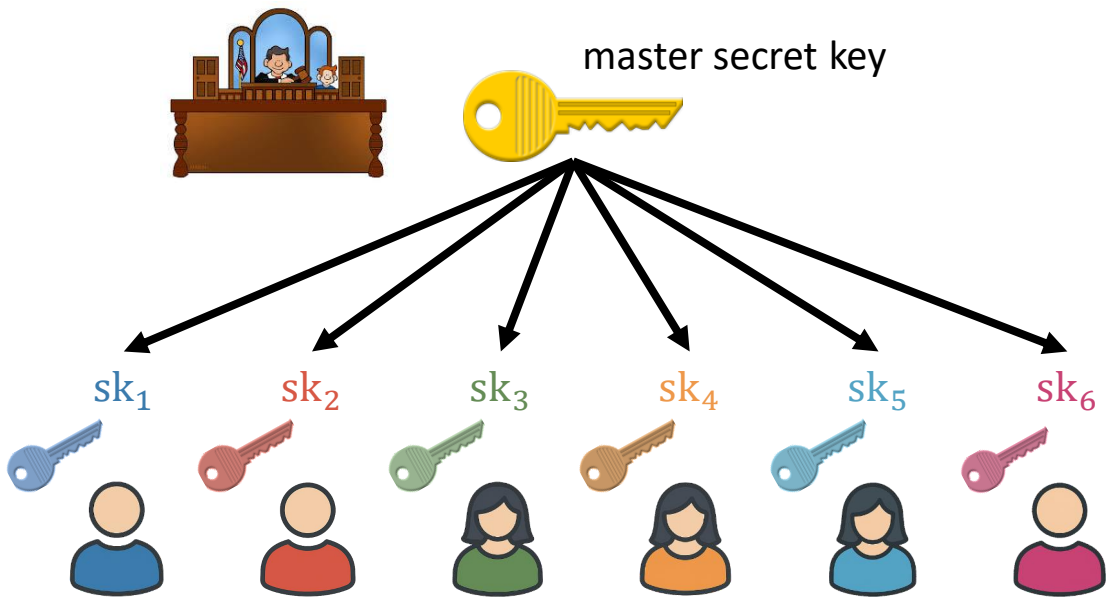
Typical setup: split a cryptographic key into many **shares**



Only an **authorized set** of parties can perform a target action (e.g., signing, decryption, etc.)

Who Generates the Shares?

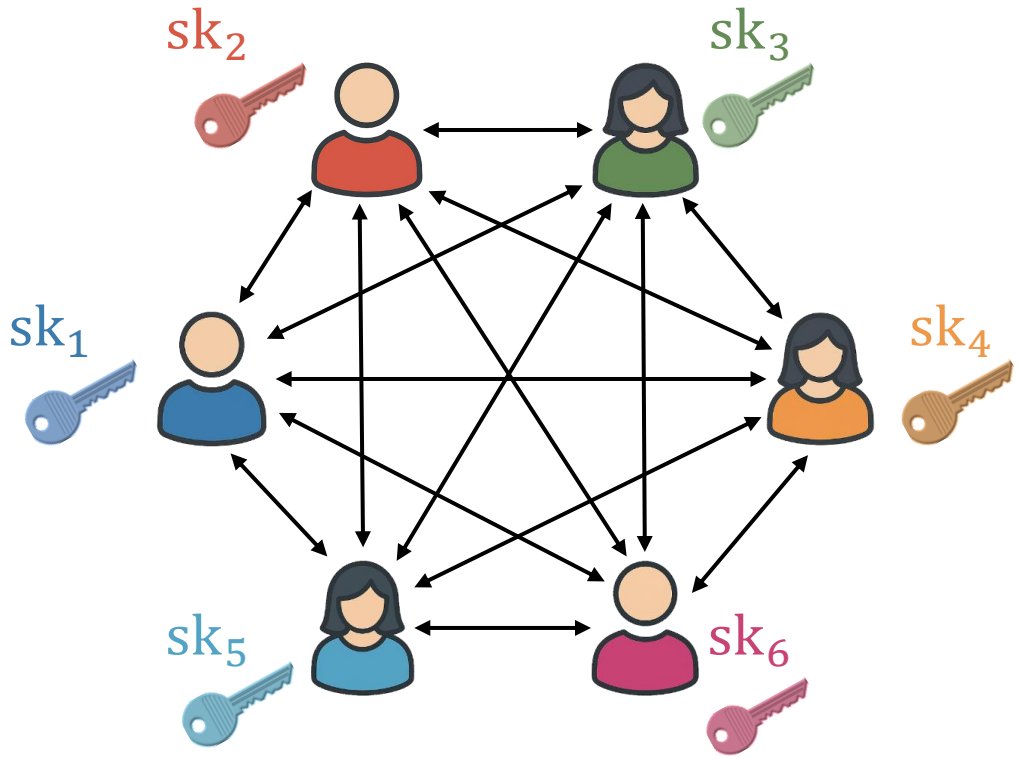
Option 1: Trusted dealer



Needs a trusted party

Redeal shares if policy changes (e.g., new user joins)

Option 2: Distributed key generation

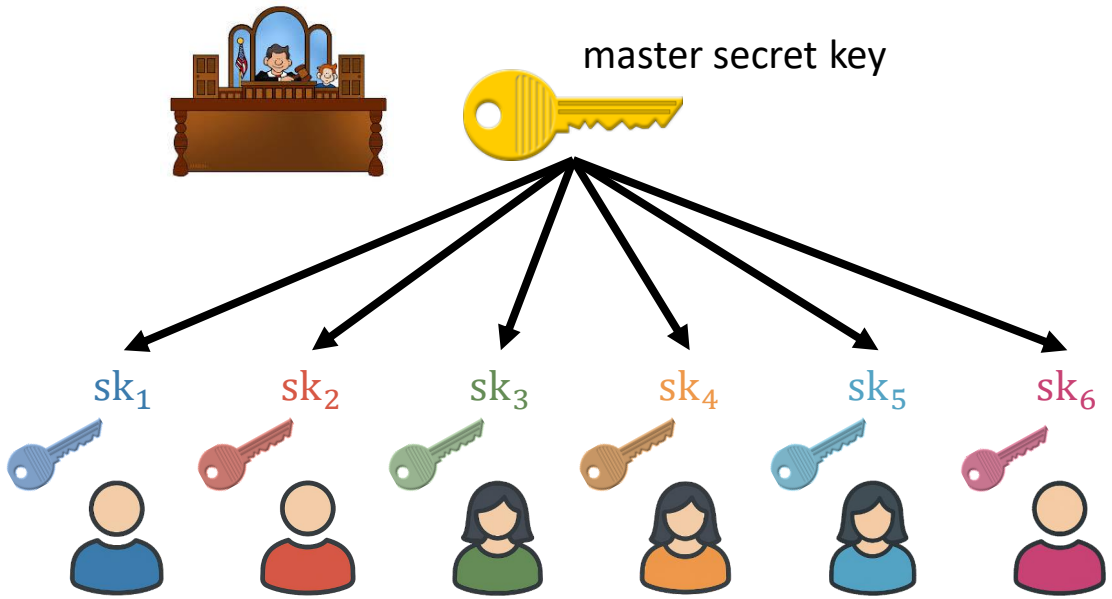


Requires parties to coordinate and interact

Rerun setup if policy changes (e.g., new user joins)

Silent Threshold Cryptography

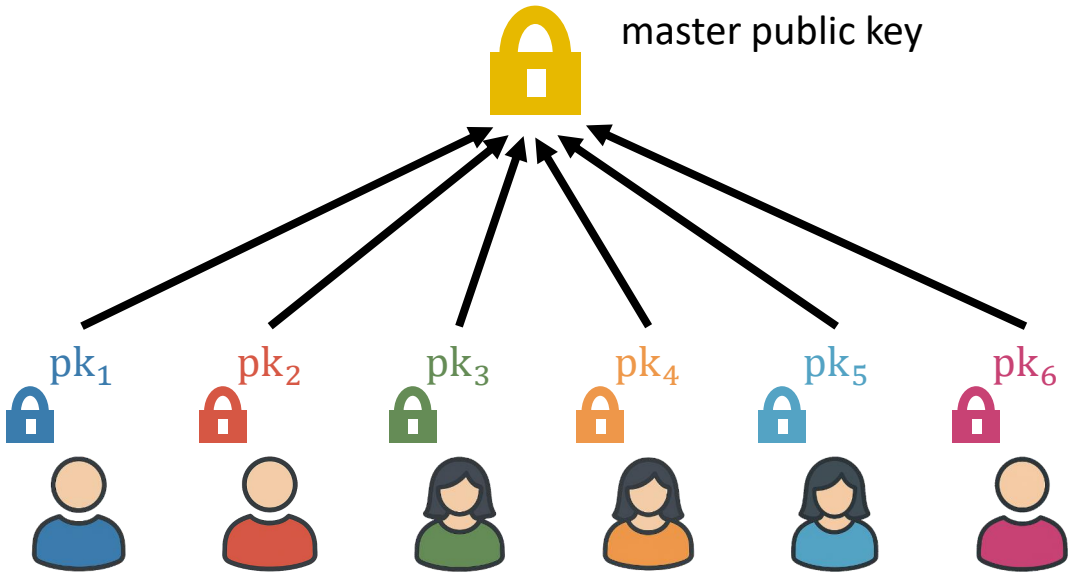
[MRVWZ21, DCXNBR23, GJMSW24]



Trusted dealer: "top-down" approach

Requires a **per-policy** setup

common reference string



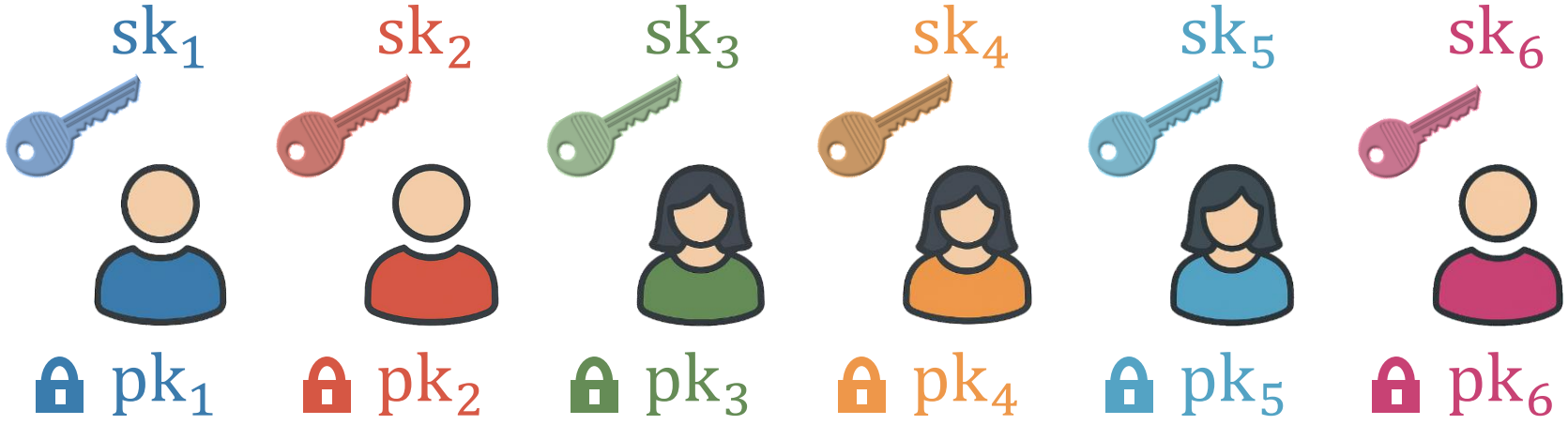
Silent setup: "bottom-up" approach

Will require a common reference string (CRS)

one-time setup

Silent Threshold Cryptography

[MRVWZ21, DCXNBR23, GJMSW24]



Users **independently** generate their own public key pk_i and secret key sk_i

---

arbitrary collection of public keys

policy

aggregated public key

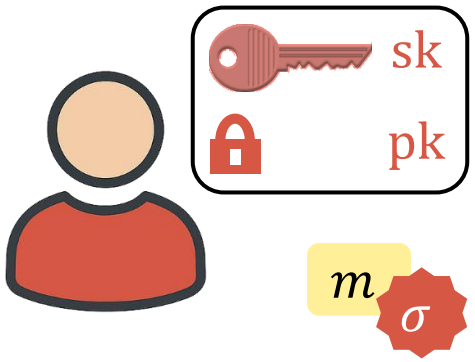
Can **deterministically** aggregate any set of public keys together with a policy

Master public key mpk serves as the key for the threshold cryptosystem for the chosen quorum

Individual secret key sk_i is each user's share

Example: Threshold Signatures with Silent Setup

[MRVWZ21, DCXNBR23, GJMSW24]



Users generate their own keys (relative to a common reference string)

$\mathrm{KeyGen}(\mathrm{crs}) \to (\mathrm{pk}, \mathrm{sk})$

Signing key sk can be used to sign messages

$\mathrm{Sign}(\mathrm{sk}, m) \to \sigma$ *(σ must verify relative to* pk*)*

arbitrary collection of public keys

policy

aggregated public key

$\mathrm{Aggregate}(\mathrm{crs}, \{\mathrm{pk}_i\}_{i \in S}, T) \to (\mathrm{mpk}, \mathrm{ht})$

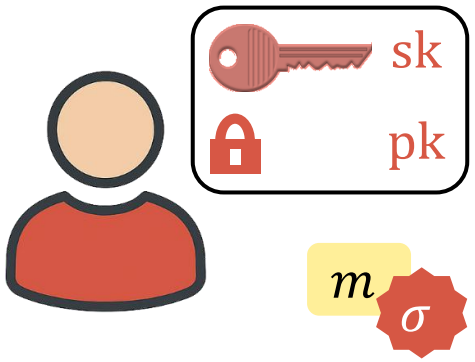
T: target threshold

mpk: aggregated verification key for quorum

ht: aggregation hint

Example: Threshold Signatures with Silent Setup

[MRVWZ21, DCXNBR23, GJMSW24]



Users generate their own keys (relative to a common reference string)

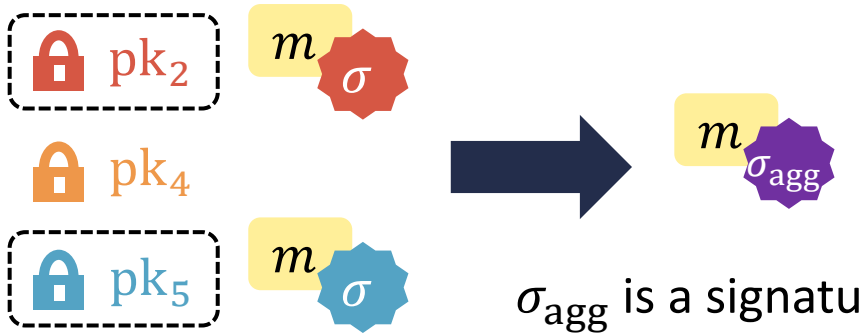
$\mathrm{KeyGen}(\mathrm{crs}) \to (\mathrm{pk}, \mathrm{sk})$

Signing key sk can be used to sign messages

$\mathrm{Sign}(\mathrm{sk}, m) \to \sigma$ *(σ must verify relative to* pk*)*

$\mathrm{Aggregate}(\mathrm{crs}, \{\mathrm{pk}_i\}_{i \in S}, T) \to (\mathrm{mpk}, \mathrm{ht})$

$\mathrm{AggSig}(\mathrm{ht}, \{\sigma_i\}_{i \in S'}) \to \sigma_{\mathrm{agg}}$



σ_{agg} is a signature on m under mpk

Efficiency: $|\sigma_{\mathrm{agg}}|$ and signature verification time are independent of number of users

Security: adversary with fewer than T signatures on m cannot forge signature with respect to mpk

$\mathrm{Verify}(\mathrm{mpk}, m, \sigma_{\mathrm{agg}}) = 1$

Silent Threshold Signatures

Scheme	Policy Family	Assumption	$\lvert\mathrm{crs}\rvert$	$\lvert\sigma\rvert$	$\lvert\sigma_{\mathrm{agg}}\rvert$
Generic (SNARK)	Boolean circuit	generic bilinear group	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$3\lvert\mathbb{G}\rvert$
Generic (BARG)	Boolean circuit	k-Lin (pairing)	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$\mathrm{poly}(\lambda)\cdot\lvert\mathbb{G}\rvert$
[DCXNBR23]	weighted threshold	generic bilinear group	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$8\lvert\mathbb{G}\rvert$
[GJMSW24]	weighted threshold	generic bilinear group	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$9\lvert\mathbb{G}\rvert + 5\lvert\mathbb{F}\rvert$

Relatively few constructions (other than via generic tools like SNARKs or BARGs)

Silent Threshold Signatures

Scheme	Policy Family	Assumption	$\lvert\text{crs}\rvert$	$\lvert\sigma\rvert$	$\lvert\sigma_{\text{agg}}\rvert$
Generic (SNARK)	Boolean circuit	generic bilinear group	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$3\lvert\mathbb{G}\rvert$
Generic (BARG)	Boolean circuit	k-Lin (pairing)	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$\text{poly}(\lambda)\cdot\lvert\mathbb{G}\rvert$
[DCXNBR23]	weighted threshold	generic bilinear group	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$8\lvert\mathbb{G}\rvert$
[GJMSW24]	weighted threshold	generic bilinear group	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$9\lvert\mathbb{G}\rvert + 5\lvert\mathbb{F}\rvert$
This work	monotone span program	q-type assumption	$O_\lambda(N^2)$	$2\lvert\mathbb{G}\rvert$	$3\lvert\mathbb{G}\rvert$
	threshold	q-type assumption	$O_\lambda(N\log N)$	$2\lvert\mathbb{G}\rvert$	$3\lvert\mathbb{G}\rvert$

Relatively few constructions (other than via generic tools like SNARKs or BARGs)

Existing constructions either have long signatures (super-constant number of group elements) or only shown secure in the generic bilinear group model

> **In fact:** all constructions with short signatures rely on some kind of SNARK machinery (e.g., sum check, inner product arguments, etc.)

This work: a direct algebraic construction (no SNARK machinery)

Silent Threshold Signatures

Scheme	Policy Family	Assumption	$\lvert\mathrm{crs}\rvert$	$\lvert\sigma\rvert$	$\lvert\sigma_{\mathrm{agg}}\rvert$
Generic (SNARK)	Boolean circuit	generic bilinear group	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$3\lvert\mathbb{G}\rvert$
Generic (BARG)	Boolean circuit	k-Lin (pairing)	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$\mathrm{poly}(\lambda)\cdot\lvert\mathbb{G}\rvert$
[DCXNBR23]	weighted threshold	generic bilinear group	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$8\lvert\mathbb{G}\rvert$
[GJMSW24]	weighted threshold	generic bilinear group	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$9\lvert\mathbb{G}\rvert + 5\lvert\mathbb{F}\rvert$
This work	monotone span program	q-type assumption	$O_\lambda(N^2)$	$2\lvert\mathbb{G}\rvert$	$3\lvert\mathbb{G}\rvert$
	threshold	q-type assumption	$O_\lambda(N\log N)$	$2\lvert\mathbb{G}\rvert$	$3\lvert\mathbb{G}\rvert$

Base signatures have two group elements, but final signature is as short as that using a pairing-based SNARK (e.g., [Gro16])

Silent Threshold Signatures

Scheme	Policy Family	Assumption	$\lvert\mathrm{crs}\rvert$	$\lvert\sigma\rvert$	$\lvert\sigma_{\mathrm{agg}}\rvert$
Generic (SNARK)	Boolean circuit	generic bilinear group	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$3\lvert\mathbb{G}\rvert$
Generic (BARG)	Boolean circuit	k-Lin (pairing)	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$\mathrm{poly}(\lambda)\cdot\lvert\mathbb{G}\rvert$
[DCXNBR23]	weighted threshold	generic bilinear group	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$8\lvert\mathbb{G}\rvert$
[GJMSW24]	weighted threshold	generic bilinear group	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$9\lvert\mathbb{G}\rvert + 5\lvert\mathbb{F}\rvert$
This work	monotone span program	q-type assumption	$O_\lambda(N^2)$	$2\lvert\mathbb{G}\rvert$	$3\lvert\mathbb{G}\rvert$
	threshold	q-type assumption	$O_\lambda(N\log N)$	$2\lvert\mathbb{G}\rvert$	$3\lvert\mathbb{G}\rvert$

Can support general policies beyond threshold policies (e.g., majority of majorities, monotone Boolean formulas)

Security in the **plain** model

Drawback: larger CRS (quadratic for general policies, quasi-linear for threshold policies)

Silent Threshold Encryption

Techniques directly generalize to setting of silent threshold public-key encryption

Scheme	Policy Family	Assumption	$\|\text{crs}\|$	$\|\text{ct}\|$
[RSY21, ADMSW24]	threshold	$i\mathcal{O}$ + OWF/SSB	None	$O_\lambda(1)$
[GKPW24]	threshold	generic bilinear group	$O_\lambda(N)$	$9\|\mathbb{G}\|$
[DJWW25]	S-space read-once TM	$i\mathcal{O}$ + SSB	$O_\lambda(1)$	$O_\lambda(2^S)$
[CW26, AGY26]	DNFs	decomposed LWE	$O_\lambda(1)$	$O_\lambda(1)$
This work	monotone span program	q-type assumption	$O_\lambda(N^2)$	$3\|\mathbb{G}\| + \|\mathbb{F}\|$
	threshold	q-type assumption	$O_\lambda(N \log N)$	$3\|\mathbb{G}\| + \|\mathbb{F}\|$

No generic SNARK-based solution in the case of encryption (except with *extractable* witness encryption)

For general policies, problem is challenging even with strong tools like witness encryption or obfuscation

This work: first construction for Boolean formulas and thresholds, but does need a large CRS

Starting Point: Boneh-Boyen Signatures

This talk: will focus just on signatures (same techniques work for encryption)

Builds on the Boneh-Boyen [BB04] pairing-based signature scheme (derived from an identity-based encryption scheme)

sk: g^{α} $(\alpha \leftarrow \mathbb{Z}_p)$

vk: $(e(g,g)^{\alpha}, u, h)$

$u, h \leftarrow \mathbb{G}$

Conventions (this talk):

- Symmetric prime-order pairing group $(\mathbb{G}, \mathbb{G}_T)$
- Group order p
- Generator g
- Pairing $e: \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$

Starting Point: Boneh-Boyen Signatures

This talk: will focus just on signatures (same techniques work for encryption)

Builds on the Boneh-Boyen [BB04] pairing-based signature scheme (derived from an identity-based encryption scheme)

sk: g^{α} $(\alpha \leftarrow \mathbb{Z}_p)$

vk: $(e(g,g)^{\alpha}, u, h)$

$u, h \leftarrow \mathbb{G}$

Sign message $m \in \mathbb{Z}_p$:

1. Sample $r \leftarrow \mathbb{Z}_p$
2. Compute "hash" of the message $u^m h$
3. Output $(g^{\alpha}(u^m h)^r, g^r)$

"encryption of the signing key g^{α} where the hash of the message $u^m h$ is the public key"

Starting Point: Boneh-Boyen Signatures

This talk: will focus just on signatures (same techniques work for encryption)

Builds on the Boneh-Boyen [BB04] pairing-based signature scheme (derived from an identity-based encryption scheme)

sk: g^{α} $(\alpha \leftarrow \mathbb{Z}_p)$

vk: $(e(g,g)^{\alpha}, u, h)$

$u, h \leftarrow \mathbb{G}$

Signature on $m \in \mathbb{Z}_p$: $(g^{\alpha}(u^m h)^r, g^r)$

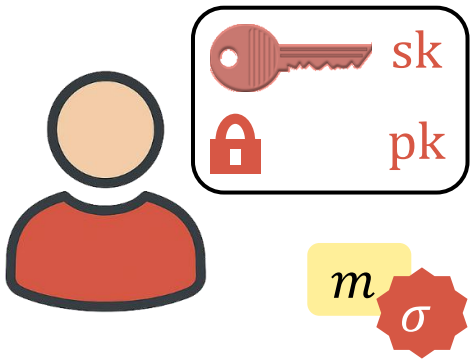
Verification: check that $e(g,g)^{\alpha} = \frac{e(g, g^{\alpha}(u^m h)^r)}{e(g^r, u^m h)}$

"decrypt in the target group via the pairing"

Construction Template

hash key

CRS: u, h



$\mathrm{sk} = \alpha$

$\mathrm{pk} = e(g,g)^{\alpha}$

$\sigma = (g^{\alpha}(u^{m}h)^{r}, g^{r})$

Each user chooses their own Boneh-Boyen public key

Signature is plain Boneh-Boyen signature

$\mathrm{pk}_1 = e(g,g)^{\alpha_1}$

$\mathrm{pk}_2 = e(g,g)^{\alpha_2}$

$\vdots$

$\mathrm{pk}_N = e(g,g)^{\alpha_N}$

Need to design two mechanisms:

1. Aggregate the user public keys relative to a policy
2. Aggregate signatures for users in the set

Aggregating Signatures

hash key

CRS: $\boxed{u, h}$

---

multi-signature scheme of [LOSSW06]

Suppose one has signature from each party $i \in S$

We will rely on linear homomorphism: $\tilde{\sigma}_i = (\overbrace{g^{\alpha_i}(u^m h)^{r_i}}^{\tilde{\sigma}_{i,1}}, \overbrace{g^{r_i}}^{\tilde{\sigma}_{i,2}})$

$$\sigma_{\text{agg},1} = \prod_{i \in S} \tilde{\sigma}_{i,1} = g^{\sum_{i \in S} \alpha_i} (u^m h)^{\sum_{i \in S} r_i} = g^{\sum_{i \in S} \alpha_i} (u^m h)^{\tilde{r}} \qquad \tilde{r} = \sum_{i \in S} r_i$$

$$\sigma_{\text{agg},2} = \prod_{i \in S} \tilde{\sigma}_{i,2} = g^{\sum_{i \in S} r_i} = g^{\tilde{r}}$$

$\sigma_{\text{agg}} = (\sigma_{\text{agg},1}, \sigma_{\text{agg},2})$ is a Boneh-Boyen signature on m with respect to $e(g,g)^{\sum_{i \in S} \alpha_i}$

Aggregating Signatures

hash key

CRS: u, h

---

multi-signature scheme of [LOSSW06]

Suppose one has signature from each party $i \in S$

We will rely on linear homomorphism: $\tilde{\sigma}_i = (\overbrace{g^{\alpha_i}(u^m h)^{r_i}}^{\tilde{\sigma}_{i,1}}, \overbrace{g^{r_i}}^{\tilde{\sigma}_{i,2}})$

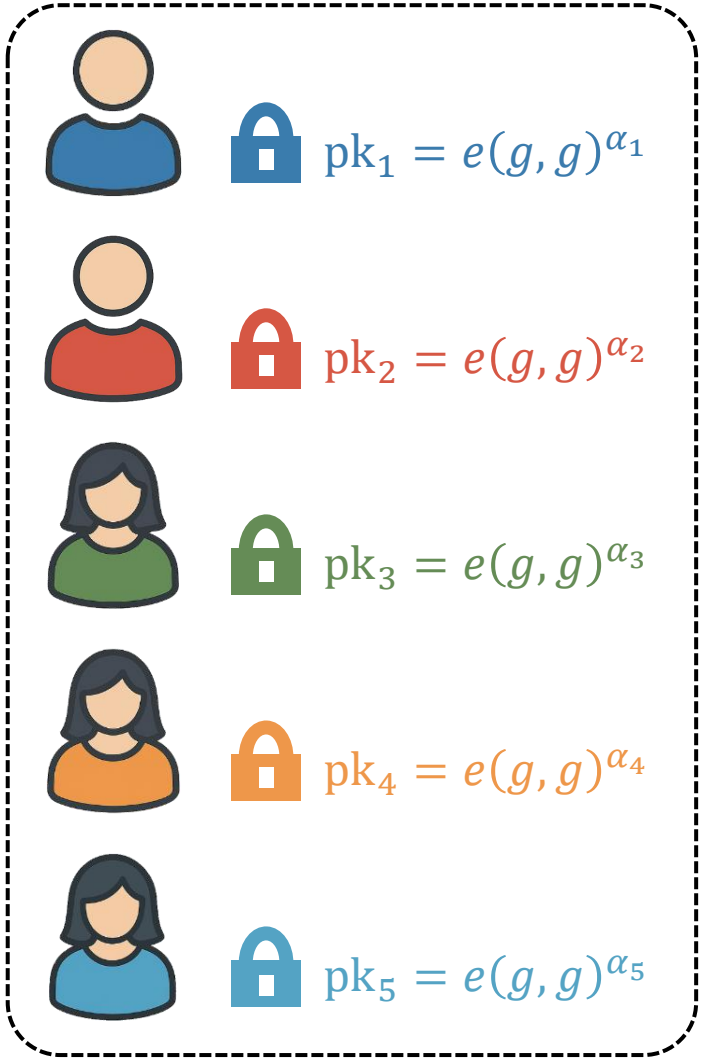
Verifier does not know the set S!

Aggregate signature $(\sigma_{\text{agg},1}, \sigma_{\text{agg},2})$ verifies with respect to $\prod_{i \in S} e(g,g)^{\alpha_i}$

Inefficient approach: Aggregate signature includes description of S so verifier can check that S satisfies the policy, and if so, compute the set-specific verification key $\prod_{i \in S} e(g,g)^{\alpha_i}$ and check the signature

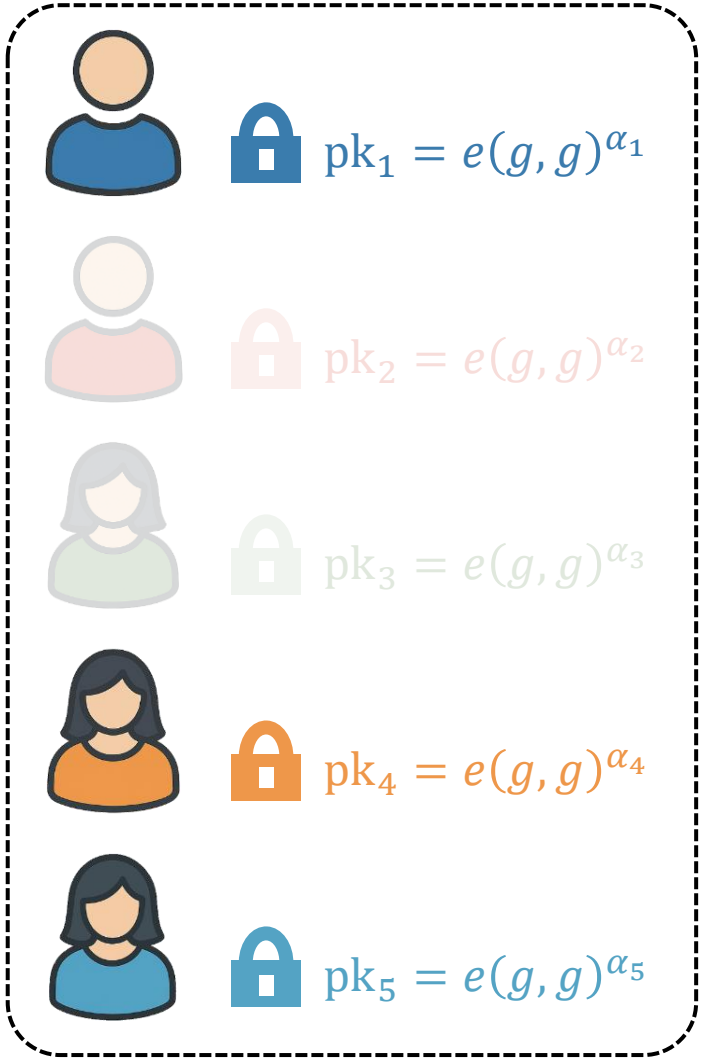
Our approach: Derive set-specific key $e(g,g)^{\sum_{i \in S} \alpha_i}$ from the pairing implicitly (from aggregated public key associated with the whole group)

Aggregating Public Keys



aggregated public key mpk
for the **entire group**

Aggregating Public Keys

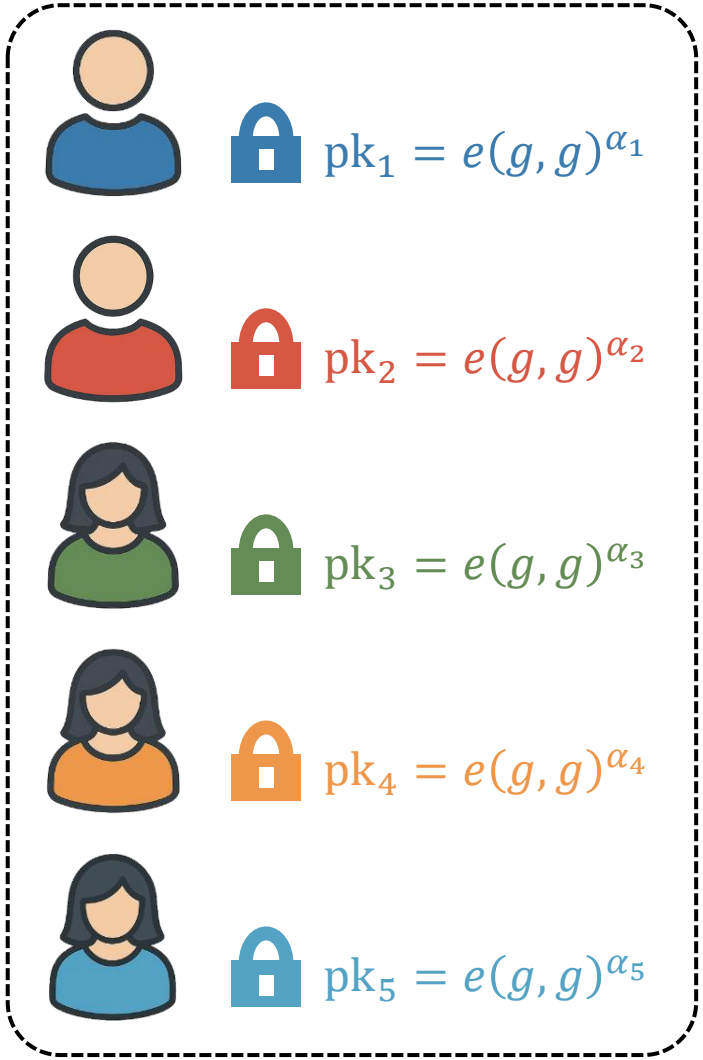


aggregated public key mpk
for the **entire group**

Goal: give out a group element to **project** mpk to the verification key for a subset S

$$\mathrm{mpk}_S = e(g,g)^{\sum_{i \in S} \alpha_i}$$

Aggregating Public Keys



aggregated public key mpk for the **entire group**

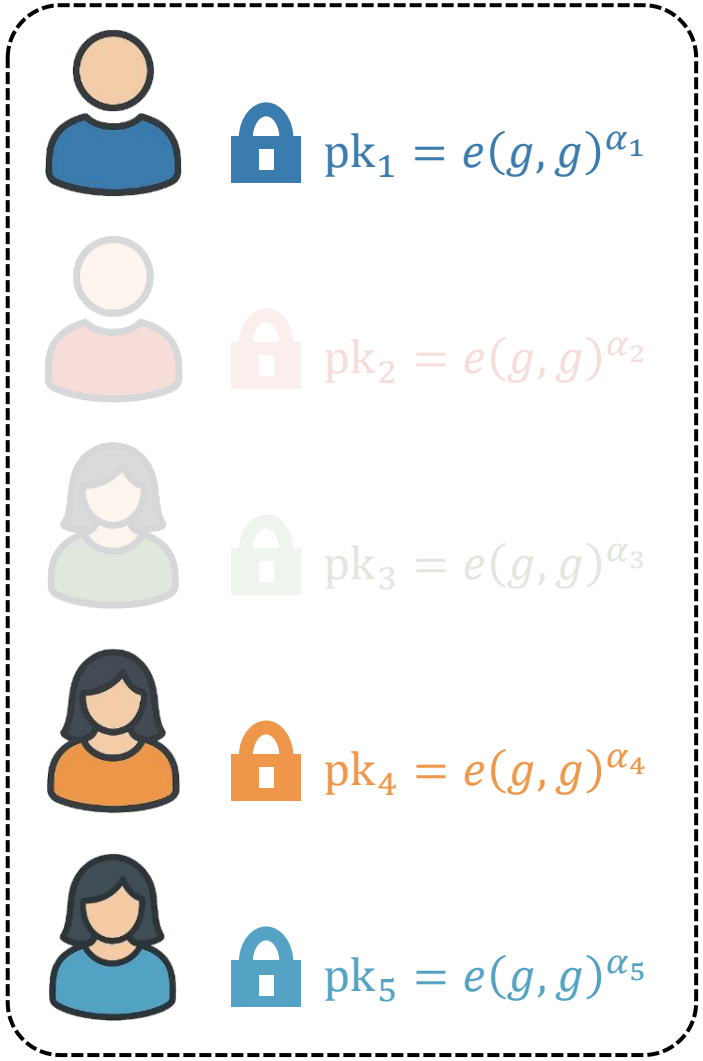
$$\mathrm{mpk} = g^{\sum_{i\in[N]} c_i \alpha_i}$$

$c_1, \ldots, c_N$ are random exponents (publish g^{c_i} in the CRS)

user's public key will include $g^{\alpha c_i}$ for all $i \in [N]$

vector commitment to $(\alpha_1, \ldots, \alpha_N)$

Aggregating Public Keys



$$\mathrm{mpk} = g^{\sum_{i\in[N]} c_i \alpha_i}$$

aggregated public key mpk for the **entire group**

Goal: give out a group element to **project** mpk to the verification key for a subset S

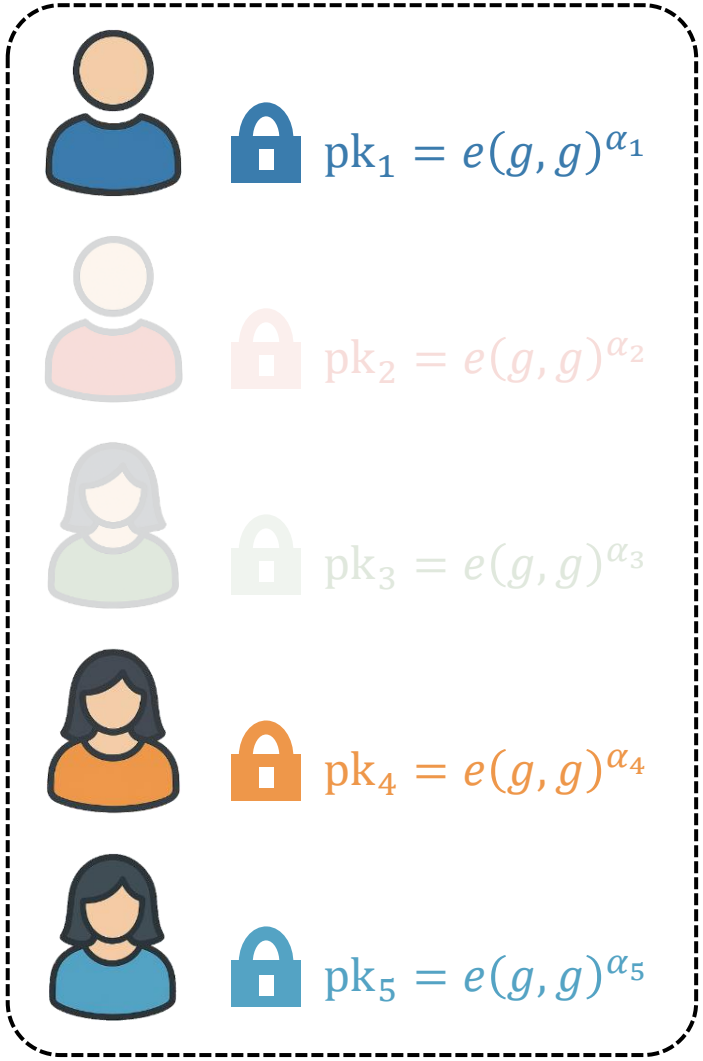
$$\mathrm{mpk}_S = e(g,g)^{\sum_{i\in S} \alpha_i}$$

To support projection, also give out g^{1/c_i} in the CRS

$$e\left(\mathrm{mpk}, g^{\sum_{i\in S} 1/c_i}\right)$$

set selector

Aggregating Public Keys



$\mathrm{mpk} = g^{\sum_{i\in[N]} c_i\alpha_i}$

aggregated public key mpk for the **entire group**

Goal: give out a group element to **project** mpk to the verification key for a subset S

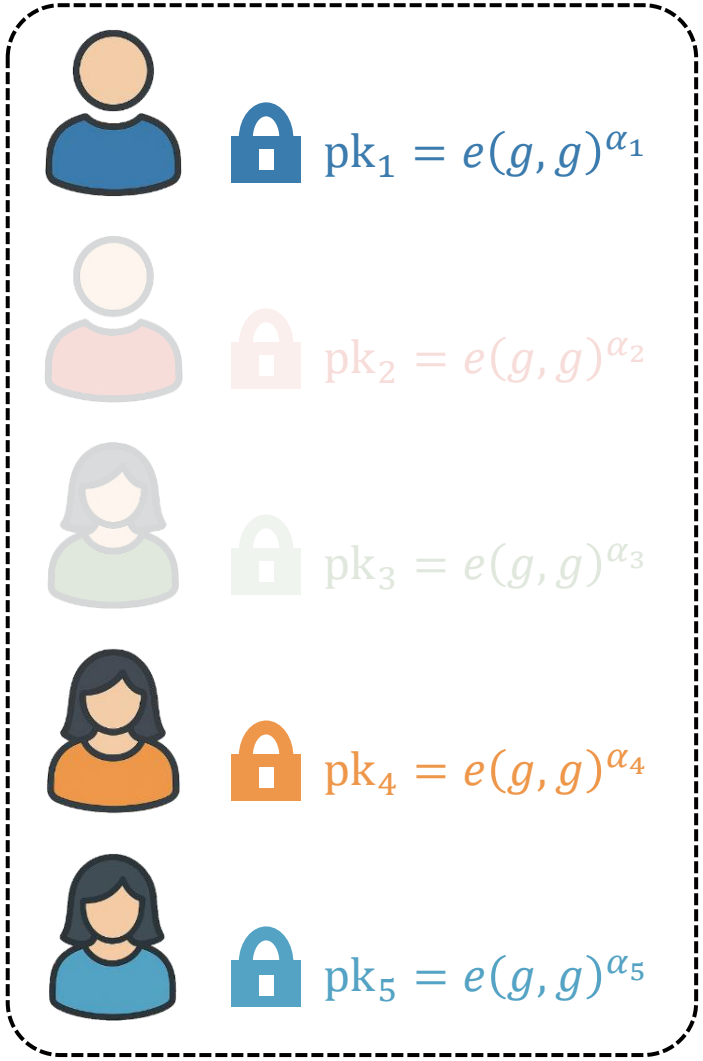
$\mathrm{mpk}_S = e(g,g)^{\sum_{i\in S}\alpha_i}$

c_i and $1/c_i$ cancel out

$e\left(\mathrm{mpk}, g^{\sum_{i\in S} 1/c_i}\right)$

set selector

Aggregating Public Keys



$$\mathrm{mpk} = g^{\sum_{i\in[N]} c_i\alpha_i}$$

aggregated public key mpk for the **entire group**

Goal: give out a group element to **project** mpk to the verification key for a subset S

c_i and $1/c_i$ cancel out

$$\mathrm{mpk}_S = e(g,g)^{\sum_{i\in S}\alpha_i}$$

$$e\left(\mathrm{mpk}, g^{\sum_{i\in S} 1/c_i}\right) = e(g,g)^{\sum_{i\in S}\alpha_i} \cdot e(g,g)^{\sum_{i\in[N]}\sum_{j\in S}\alpha_i c_i/c_j}$$

set selector — target quantity — cross terms

Aggregating Public Keys

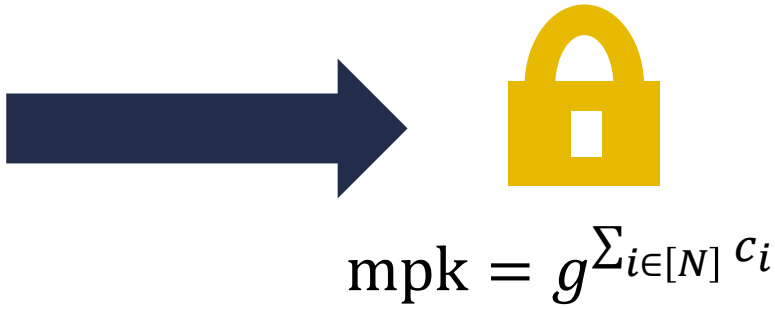
CRS contains:
- commitment components g^{c_i}
- selector components g^{1/c_i}
- cross-terms g^{c_i/c_j}

Aggregate signature contains:
- set selector $z_1 = g^{\sum_{i \in S} 1/c_i}$
- cross term $z_2 = g^{\sum_{i \in [N]} \sum_{j \in S} \alpha_i c_i / c_j}$
- aggregated Boneh-Boyen signature (σ_1, σ_2)

Verification:
- compute projected verification key

$$e(g,g)^{\sum_{i \in S} \alpha_i} = \frac{e(\mathrm{mpk}, z_1)}{e(g, z_2)}$$

- verify (σ_1, σ_2) with respect to $e(g,g)^{\sum_{i \in S} \alpha_i}$



$$\mathrm{mpk} = g^{\sum_{i \in [N]} c_i \alpha_i}$$

aggregated public key mpk for the **entire group**

Goal: give out a group element to **project** mpk to the verification key for a subset S

c_i and $1/c_i$ cancel out

$$\mathrm{mpk}_S = e(g,g)^{\sum_{i \in S} \alpha_i}$$

$$e\left(\mathrm{mpk}, g^{\sum_{i \in S} 1/c_i}\right) = e(g,g)^{\sum_{i \in S} \alpha_i} \cdot e(g,g)^{\sum_{i \in [N]} \sum_{j \in S} \alpha_i c_i / c_j}$$

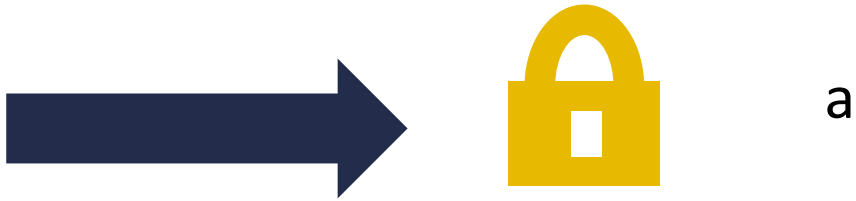
set selector | target quantity | cross terms

Aggregating Public Keys

CRS contains:
- commitment components g^{c_i}
- selector components g^{1/c_i}
- cross-terms g^{c_i/c_j}

Aggregate signature contains:
- set selector $z_1 = g^{\sum_{i \in S} 1/c_i}$
- cross term $z_2 = g^{\sum_{i \in [N]} \sum_{j \in S} \alpha_i c_i / c_j}$
- aggregated Boneh-Boyen signature (σ_1, σ_2)

Verification:
- compute projected verification key
$$e(g,g)^{\sum_{i \in S} \alpha_i} = \frac{e(\mathrm{mpk}, z_1)}{e(g, z_2)}$$
- verify (σ_1, σ_2) with respect to $e(g,g)^{\sum_{i \in S} \alpha_i}$



$$\mathrm{mpk} = g^{\sum_{i \in [N]} c_i \alpha_i}$$

aggregated public key mpk for the **entire group**

Goal: give out a group element to **project** mpk to the verification key for a subset S

$$\mathrm{mpk}_S = e(g,g)^{\sum_{i \in S} \alpha_i}$$

Still need to certify that S satisfies the access policy!
Short answer: embed an "interpolation check" in public key

See paper for details!

Summary

Scheme	Policy Family	Assumption	$\lvert\mathrm{crs}\rvert$	$\lvert\sigma\rvert$	$\lvert\sigma_{\mathrm{agg}}\rvert$
Generic (SNARK)	Boolean circuit	generic bilinear group	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$3\lvert\mathbb{G}\rvert$
Generic (BARG)	Boolean circuit	k-Lin (pairing)	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$\mathrm{poly}(\lambda)\cdot\lvert\mathbb{G}\rvert$
[DCXNBR23]	weighted threshold	generic bilinear group	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$8\lvert\mathbb{G}\rvert$
[GJMSW24]	weighted threshold	generic bilinear group	$O_\lambda(N)$	$\lvert\mathbb{G}\rvert$	$9\lvert\mathbb{G}\rvert + 5\lvert\mathbb{F}\rvert$
This work	monotone span program	q-type assumption	$O_\lambda(N^2)$	$2\lvert\mathbb{G}\rvert$	$3\lvert\mathbb{G}\rvert$
	threshold	q-type assumption	$O_\lambda(N \log N)$	$2\lvert\mathbb{G}\rvert$	$3\lvert\mathbb{G}\rvert$

Take-away: algebraic framework yields schemes for **general policies** with **short aggregate signatures** and **security in the plain model**

Cost: **larger CRS** (for dynamic threshold policies, difference is quasilinear vs. strictly linear)

Open Problems

Pairing-based schemes with transparent setup (and comparable signature/ciphertext size)

Silent threshold cryptography from post-quantum cryptographic assumptions

Thanks!

https://eprint.iacr.org/2025/1547.pdf